マークシート選択式試験作成における数式処理の活用 (数学ソフトウェアとその効果的教育利用に関する研究)

AUTHOR(S):

濱田, 龍義; 中川, 義行

マークシート選択式試験作成における数式処理の活用

日本大学・生物資源科学部　濱田　龍義
Tatsuyoshi Hamada, College of Bioresource Sciences, Nihon University
龍谷大学・経済学部　中川　義行
Yoshiyuki Nakagawa, Faculty of Economics, Ryukoku University

1　はじめに

本稿は印刷物によるマークシート選択式試験作成における数式処理活用の報告である．マークシート選択式試験問題作成には Auto Multiple Choice [1]（以下，AMC）を用いる．AMC はフランスの Alexis Bienvenüe(ISFA: Institut de Science Finacière et d'Assurances) が中心になって開発を進めている GPLv2 のオープンソースソフトウェアである．TEX, Perl, OpenCV, C++ 等で開発されており，岡山大学の籠谷裕人によって日本語ドキュメントも整備されている．利用には Linux, FreeBSD 等の Unix 系システムが簡単であるが，MacOS 上のパッケージ管理システム MacPorts や，Windows 上で Linux 環境を実現する WSL(Windows Subsystem for Linux) 等にも対応している．本稿では，Debian GNU/Linux buster (stable) で動作確認を行った．

AMC は，選択肢の無作為並替えや複数種類の問題用紙作成が可能である．また，グループ化された設問の順序の並べ替えと抽出設問数の指定などもできる．ここでは，AMC の利用の流れを簡単に紹介し，数式処理システムとの連携方法について述べる．AMC の詳しい利用方法については [2] や，AMC ドキュメント等を参照されたい．

2　AMC の利用

AMC は Debian GNU/Linux の標準パッケージに含まれているので，管理者権限で

```
apt install auto-multiple-choice
```

を実行することにより関連パッケージも含めてインストールできる．デスクトップのメニューから [教養・教育]→[Auto Multiple Choice] を選択するか，端末に

```
auto-multiple-choice
```

を入力することで起動できる．

AMC を利用して選択式試験を行う際は，AMC の起動から『試験問題作成』→ 試験の実施 → 答案の PDF 化 →『マーク認識』→『採点』→『レポート』という手順となる．ここで，『』は AMC のタブメニュー，「」は AMC のボタンメニューを表す．また，以降の記述で 〔〕 は AMC のツリーメニューを表すことにする．

2.1 AMC の起動と試験問題の作成

AMC を起動すると，図 1 のようなメインインターフェースが表示される．プロジェクト作成前なので，4 つのタブメニュー『試験問題作成』,『マーク認識』,『採点』,『レポート』は薄く表示されている．左から右へ順番に作業を行うことになる．

上部左から 2 番目のボタン をクリックすると図 2 のように新規 AMC プロジェクトを作成できる．ここでは仮にプロジェクト名を 1st_project としておく．

図 1: AMC 起動画面

図 2: 新規 AMC プロジェクトを作成

新規プロジェクトを作成すると，図 3 のように LaTeX ソースの選択画面が表示される．初めての時はテンプレートを選択すると良い．図 4 に表示されているように日本語のテンプレートも用意されている．ここでは〔[JA] ドキュメント〕の〔単純な例〕を選択することにする．なお，LaTeX に慣れていない方のために，マークアップ言語の一種である AMC-TXT 形式のテンプレートも用意されている．

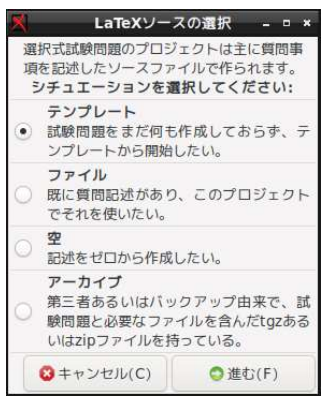

図 3: テンプレートを選択

図 4: 単純な例を選択

新規プロジェクトを作成してテンプレートを指定したが未編集の状態が図 5 である．「ソースファイル編集」ボタンをクリックすると，標準で設定されているテキストエディタが起動する．もし，別のテキストエディタを選択したいときは最上部にあるボタンをクリックすると変更できる．現在，日本語 LaTeX 処理系としては，pLaTeX 2_ε, pdfLaTeX+BXjscls, LuaTeX-ja 等の複数の選択肢がある．本稿では LuaTeX-ja を用いる．ソースコードに埋め込まれた Lua スクリプトを用いて外部の数式処理システムを起動するため，デフォルト LaTeX エンジンを `lualatex --shell-escape` に変更する[1]．また，テンプレートで用意されたソースコードの変更が必要である．ここでは

```
\documentclass[a4paper]{ltjsarticle}
```

に変更する．

ソースコードを変更後に「文書更新」ボタンをクリックして LaTeX タイプセットを行う．無事にタイプセットが終わると図 6 の状態になり，作業文書最終更新が緑色に変化する．「文書更新」では「文書」→「設問カタログ」,「模範解答」,「個別問題回答」なども更新される．

図 5: 新規作成して未編集の状態

図 6: 「文書更新」後

最後に「レイアウト検出」ボタンをクリックしてマーク位置を特定した状態が図 7 である．複数の試験問題に対して，マーク位置が特定される．

なお，試験問題の作成と実施にあたっては以下の 2 種類の方法が存在する．

1. 受験者全員分の試験問題用紙を別々の試験問題番号で作成し印刷する．
2. 少数の試験問題（場合によっては 1 種類）だけを印刷し，複製によって受験者全員分の試験問題用紙を印刷する．この場合は，試験問題用紙は 1 枚だけに限るという制限が生じる．AMC は同じ受験者が記入した 2 つのページを結びつけることができない．

本稿では，2 番目の少数の試験問題を複製して配布する場合を想定している．

[1] 外部命令の実行はセキュリティ上の問題が起こることもあるので注意が必要である．

図 7: 「レイアウト検出」後

図 8: マーク認識

2.2 試験の実施と答案用紙の PDF 化

試験問題を印刷し学生に配布したら，それ以降は作業文書を変更することは出来ない．これは，印刷物のマーク認識の問題であり完全に同一しておく必要があるためである．

答案用紙を回収したらスキャナ等で PDF 化を行う．PDF 化に対応しているコピー機がある時はコピー機を用いると良い．対応していない場合は安価に入手可能なシートフィーダスキャナを利用する．

答案用紙を PDF 化したファイルを用意したら，AMC のタブメニューから図 8 のように『マーク認識』を選択し,「自動」ボタンをクリックして PDF 化したファイルを選択する．この際に，図 9 のように，答案用紙モードの選択が必要がある．ここでは全員が異なる問題を解く必要はないと考え，2 番目の〔コピー答案用紙モード〕を選択することにする．マーク認識は図 10 のように個別に用紙を選択して確認することもできる．この際，誤認識している場合は「手動」で修正を行う．

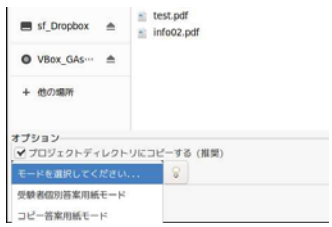

図 9: 答案用紙モードの選択

図 10: 用紙を選択して「拡大画像」

2.3 『採点』

マークが正しく認識されたら『採点』タブメニューに移動する.「採点」ボタンをクリックすると答案の採点が行われる．なお，事前に受験者名簿を CSV ファイルで登録しておけば，マークによる受験者自動識別も可能である.

図 11: 採点前の状態

図 12: 採点後の状態

2.4 『レポート』

採点が終わったら,『レポート』タブメニューから,〔採点結果のエクスポート〕を行うことができる．CSV，PDF, OpenOffice 形式などの複数のデータ形式にエクスポートすることができる．図 13 は，OpenOffice 形式にエクスポートした例である．また,〔答案への採点記入〕を行うことで受験者 1 人に 1 ファイル，もしくは受験者全員で 1 ファイルのレポートを作成できる．事前に電子メールを記入した受験者名簿を作成しており，かつ，受験者 1 人に 1 ファイルの答案への採点記入を行っていれば，各受験者へのメールの「送信」により，採点済みの答案を返却することもできるが，若干の設定が必要なので，AMC ドキュメントを参照されたい.

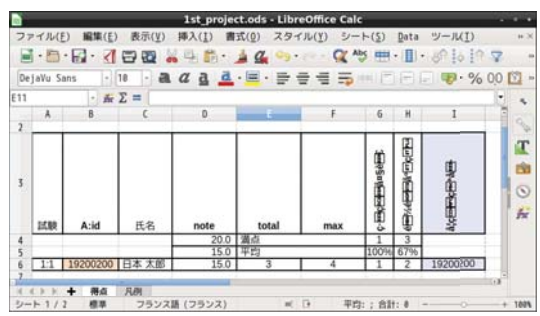

図 13: OpenOffice 形式へのエクスポート

2.5 単一正解のある選択式問題

具体的な選択式問題の作成方法は，AMC ドキュメントに詳しい．以下の例は，AMC ドキュメントに紹介されている単一正解のある選択式問題のソースコード例である．LaTeX Beamer 等のプレゼンテーション環境に慣れている方ならば，特に難しいことはないであろう．

```
\documentclass[a4paper]{ltjsarticle}
\usepackage[box,completemulti,lang=JA]{automultiplechoice}
\begin{document}
\onecopy{10}{ % 選択肢が無作為並替えされた問題用紙 10 種類が生成される．
 % ヘッダ部
  \begin{question}{総理大臣}
    日本の総理大臣になったことがある人を一人選びなさい。
    \begin{choices}
      \correctchoice{大隈重信}
      \wrongchoice{湯川秀樹}
      \wrongchoice{聖徳太子}
      \wrongchoice{徳川家康}
    \end{choices}
  \end{question}
}
\end{document}
```

2.6 任意個数正解のある選択式問題

任意個数正解のある選択式問題を作成するときは questionmult 環境を用いる．

```
\begin{questionmult}{都道府県}
  次のうち、日本の都道府県はどれか、すべて選びなさい。
  \begin{choices}
    \correctchoice{石川}
    \wrongchoice{山田}
    \correctchoice{宮崎}
  \end{choices}
\end{questionmult}
```

選択肢が短い場合には \begin{choiceshoriz}...\end{choiceshoriz} を用いて水平に並べると場所を取らない．特にコピー答案用紙モードを用いる際は，1 枚の用紙内に設問を収められるかどうかが重要なので，場合によって使い分けることになる．

3　AMC と数式処理システムの連携

LuaTEX はスクリプト言語 Lua を TEX ソースコードに埋め込むことができ，標準の io.popen() 関数を用いて外部アプリケーションと連携できる [3]．また，Lua 言語はテーブル等のデータ型を気軽に用いることができることも重要な要素である．LuaTEX-ja だけでも Lua 言語を埋め込めるが，%や\ などの扱いが面倒なので TEX マクロの luacode を読み込んでおく．

```
\usepackage{luacode}
```

Lua 言語で定義された変数を展開し，TEX 形式で受け取り表示するための LATEX ユーザ定義命令を\newcommand で作成する．

```
\newcommand*{\var}[1]{\luaexec{tex.print(#1)}}
```

数式処理システム Maxima の命令を文字列として受け取り，数式処理結果を TEX 形式で戻す関数 execMaxima() を作成する．

```
\begin{luacode*}
function execMaxima(cmd)
  local texcmd="echo 'tex1("..cmd..");' | maxima --very-quiet"
  local hdl=io.popen(texcmd, "r")
  local content=string.gsub(hdl:read("*all"), "\n", "")
  hdl:close()
  return content
end
\end{luacode*}
```

AMC のソースコード内で execMaxima() を用いて正答や誤答を作成することができる．もし，Maxima 以外の数式処理システムを用いる場合には，texcmd の部分を書き換えれば良い．Sage, Maple, Wolfram Engine の例を紹介しておく．

```
texcmd="sage -c 'print(latex("..cmd.."))'"
```

```
texcmd="echo 'latex("..cmd..");'|maple -q"
```

```
texcmd="wolframscript -code 'TeXForm["..cmd.."]'"
```

3.1　作成例

以下に偏導函数に関する例題を紹介する．ソースコード中で係数 a, b, s, t は Lua 言語の函数 math.random() で乱数を生成している．Lua 言語における文字列の結合は終止

符 2 個を用いる．また，exp や diff は Maxima の函数である．

```
\begin{question}{pdiff03}
\luadirect{
  a=math.random(2, 9);
  b=math.random(2, 9);
  s=(-1)^(math.random(0, 1));
  t=(-1)^(math.random(0, 1));
  g=s*a..'*x+'..t*b..'*y';
  g1=s*(a-1)..'*x+'..t*b..'*y';
  g2=s*a..'*x+'..t*(b-1)..'*y';
  f='exp('..g..')';
  f1='exp('..g1..')';
  f2='exp('..g2..')';
  formula=execMaxima(f);
  correct1=execMaxima('diff('..f..', x)');
  wrong1=execMaxima('diff('..f1..', x)');
  wrong2=execMaxima('diff('..f2..', x)');
  wrong3=execMaxima('diff('..f..', x)/'..a*s);
  wrong4=execMaxima('diff('..f1..', x)/'..a*s);
}

函数 \(f(x, y)=\var{formula}\) の偏導函数\(f_{x}\) を求めなさい．
  \begin{choiceshoriz}
    \correctchoice{\(\var{correct1}\)}
    \wrongchoice{\(\var{wrong1}\)}
    \wrongchoice{\(\var{wrong2}\)}
    \wrongchoice{\(\var{wrong3}\)}
    \wrongchoice{\(\var{wrong4}\)}
  \end{choiceshoriz}
\end{question}
```

以下が上記ソースコードの実行結果である．標準ではマークの形状は長方形だが，TEX マクロの TikZ を用いることで長円形に変更することもできる．

問 3

函数 $f(x,y)=e^{9x-2y}$ の偏導函数 f_x を求めなさい．

○ $9e^{9x\ \ 2y}$　○ $8e^{8x\ \ 2y}$　○ $e^{9x\ \ 2y}$　○ $\frac{8e^{8x-2y}}{9}$　○ $9e^{9x\ \ y}$

4 まとめ

Lua 言語標準の関数 `io.popen()` を用いることで，TEX ソースコード内から数式処理システムを簡単に呼び出すことができるようになった．乱数を用いて設問自動生成を行い，学生個別に異なる問題に取り組ませることができる．講義中の演習時間では，学生同士が相談して問題に取り組むことも許しているが，それぞれ異なる問題を出題することで，単なる丸写しとなることを防ぐこともできる．数式処理システムと Lua のテーブルを利用することにより，係数の乱数化に留まらない運用も可能である．

今回は `io.popen()` を用いたが，数式処理システムのプロセスを毎回起動するので，計算機の負荷が高い．ソケットを用いる方法 [4] や，FFI を用いる方法 [5] などについても検討されており，引き続き検証を進める予定である．なお，数式処理命令の入力方法についても課題が多い．例えば，函数の導函数を求めるという基本的な命令でも，数式処理システムごとに微妙に異なることがある．TEX に埋め込まれた数式を自動変換して数式処理に渡すシステムについては [6] があるが，今後も検討が必要な部分と思われる．

現在は経験と試行錯誤，数式処理システムの支援によって選択肢を作成しているが，教育効果を期待できる選択肢を作成することは難しい問題である．今後の数式処理システムの応用の 1 つとして期待できる領域ではないかと考えている．

参考文献

[1] Alexis Bienvenüe, `https://www.auto-multiple-choice.net/`

[2] マークシート選択式問題における数式処理の活用，濱田龍義，中川義行，日本数式処理学会第 28 回大会，`https://www.slideshare.net/hamadatatsuyoshi/ss-148398845`

[3] lua で linux のコマンドを実行する, hidetzu, `https://qiita.com/hidetzu/items/623cec2ec171db57c246`

[4] ソケットを使って Lua ／ LuaTeX 文書から Maxima を呼び出す，mod_poppo, `https://qiita.com/mod_poppo/items/6a6c83cb8cd9dbd73627`

[5] LuaLaTeX の FFI を使って Pari ライブラリを呼び出す，木村巌, `https://qiita.com/iwaokimura/items/4ac945c8e8897a029a61`

[6] 数式処理を用いた TeX 文書への数式の解の自動挿入システムの作成，黒木浩聖，指導教員 大墨礼子, `http://www.salesio-sp.ac.jp/papers/sotsuken/2015/html/cs/`，2015 年度サレジオ工業高等専門学校特別研究・卒業研究概要集, 5402, 2015,

[7] 理科系の基礎 微分積分, 高遠節夫, 石村隆一, 野田健夫, 安冨真一, 山方竜二, 培風館, 2013.